SWEN7302
Secure Software Development

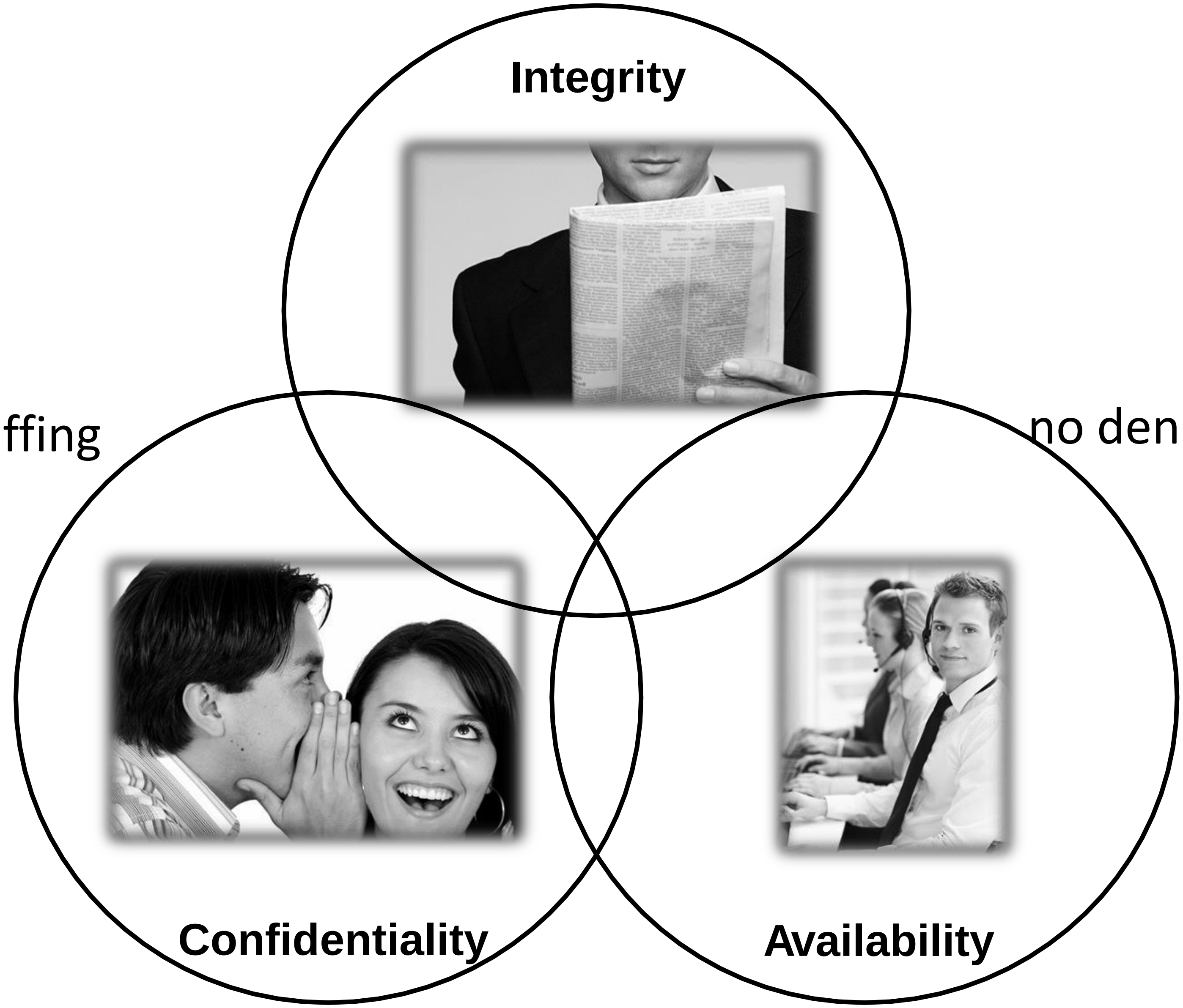
Secure Software Concepts

Sep 23, 2017

What is Security?

- Security is the *prevention* of certain types of *intentional* actions from occurring in a system.
  - The actors who may attack a system are ***threats***.
  - Threats carry out ***attacks*** to compromise a system.
  - Objects of attacks are ***assets***.

Components of Security



no man-in-the-middle attacks

Integrity

no sniffing

no denial of service

Confidentiality

Availability

Confidentiality

- **Confidentiality** is the avoidance of the unauthorized disclosure of information.

- Examples where confidentiality is critical:
  - Personal information
  - Trade secrets
  - Military plans

Security Controls for Confidentiality

- **Access Control:** rules and policies that limit access to certain people and/or systems.
  - File permissions (which users can access)
  - Firewall settings (which IP addresses can access)
- **Encryption:** transforming information so that it can only be read using a secret key.
  - Block cipher, e.g. AES-CBC
  - Stream cipher, e.g. RC4

Integrity

- **Integrity** is the property that information has not be altered in an unauthorized way.

- Examples where integrity is critical:
  - Operating system files
  - Software updates and downloads
  - Bank account records

Security Controls for Integrity

- **Cryptographic Checksums:**
  - The computation of a function that maps the contents of a file to a numerical value.
  - Hash function, e.g. SHA-256, SHA-3
  - Message authentication code (MAC) for data authentication and integrity, e.g. HMAC-SHA256
- **Intrusion detection:**
  - Systems that look for signatures of attacks or that verify that all system software matches correct checksums.
- **Backups:** periodic archiving of data.

Availability

- **Availability** is the property that information is accessible and modifiable in a timely fashion by those authorized to do so.

- Examples where availability is critical:
  - E-commerce site
  - Authentication server for your network
  - Current stock quotes

Security Controls for Availability

- **Physical protections:** infrastructure meant to keep information available even in the event of physical challenges.
  - Backup generators
  - Disaster recovery site

- **Computational redundancies:** computers and storage devices that serve as fallbacks in the case of failures.
  - Backup tapes
  - RAID

Other Security Components

no spoofing

Authenticity

non-Repudiation

Signatures

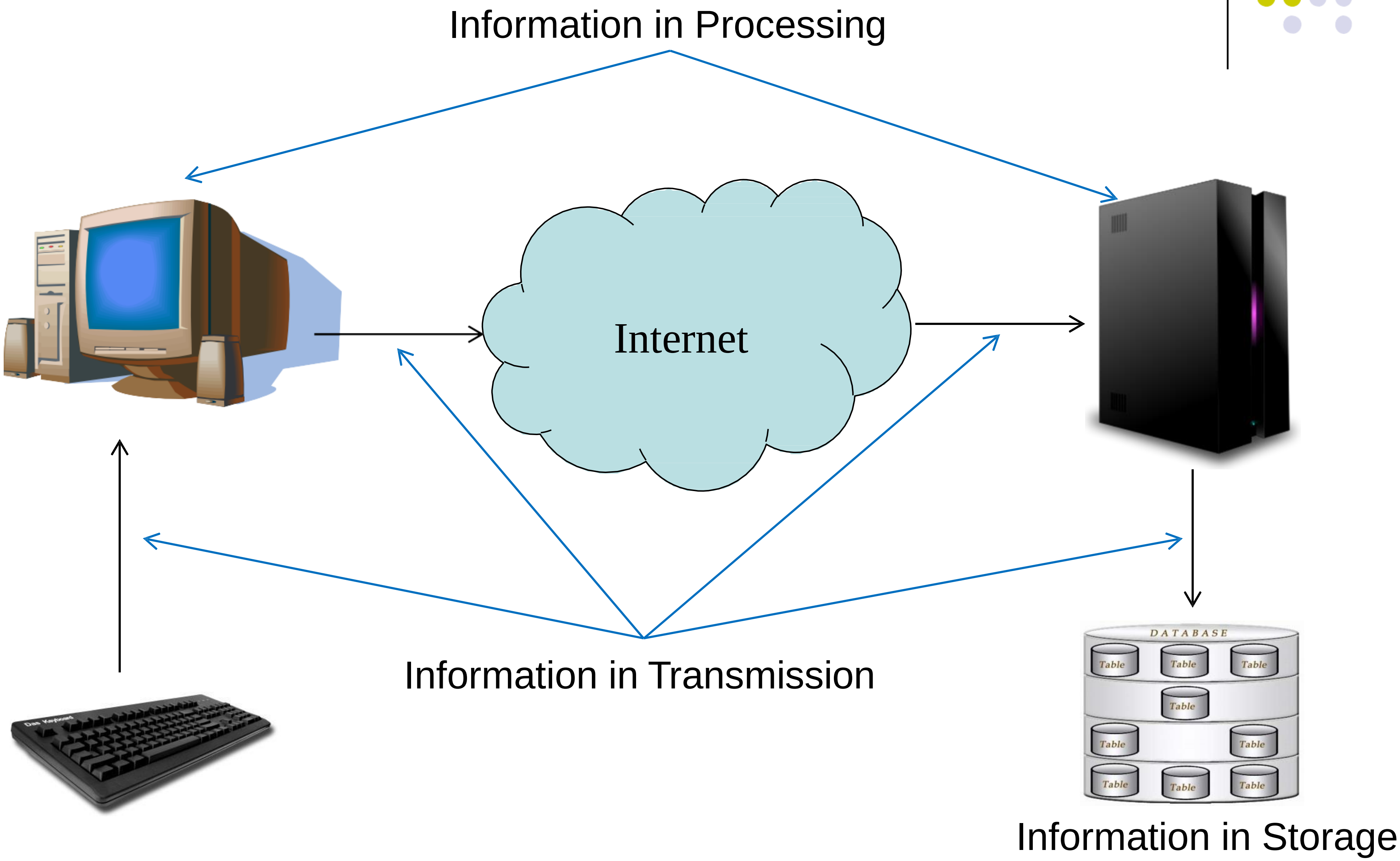
Privacy

no traffic analysis or location tracking

States of Information

1. **Storage**
   - information in permanent storage (disk or tape) that is not currently being accessed
2. **Processing**
   - information in memory (RAM or cache) that is currently being used by a program
3. **Transmission:**
   - information in transit between one node and another on a network

Securing Information in All States



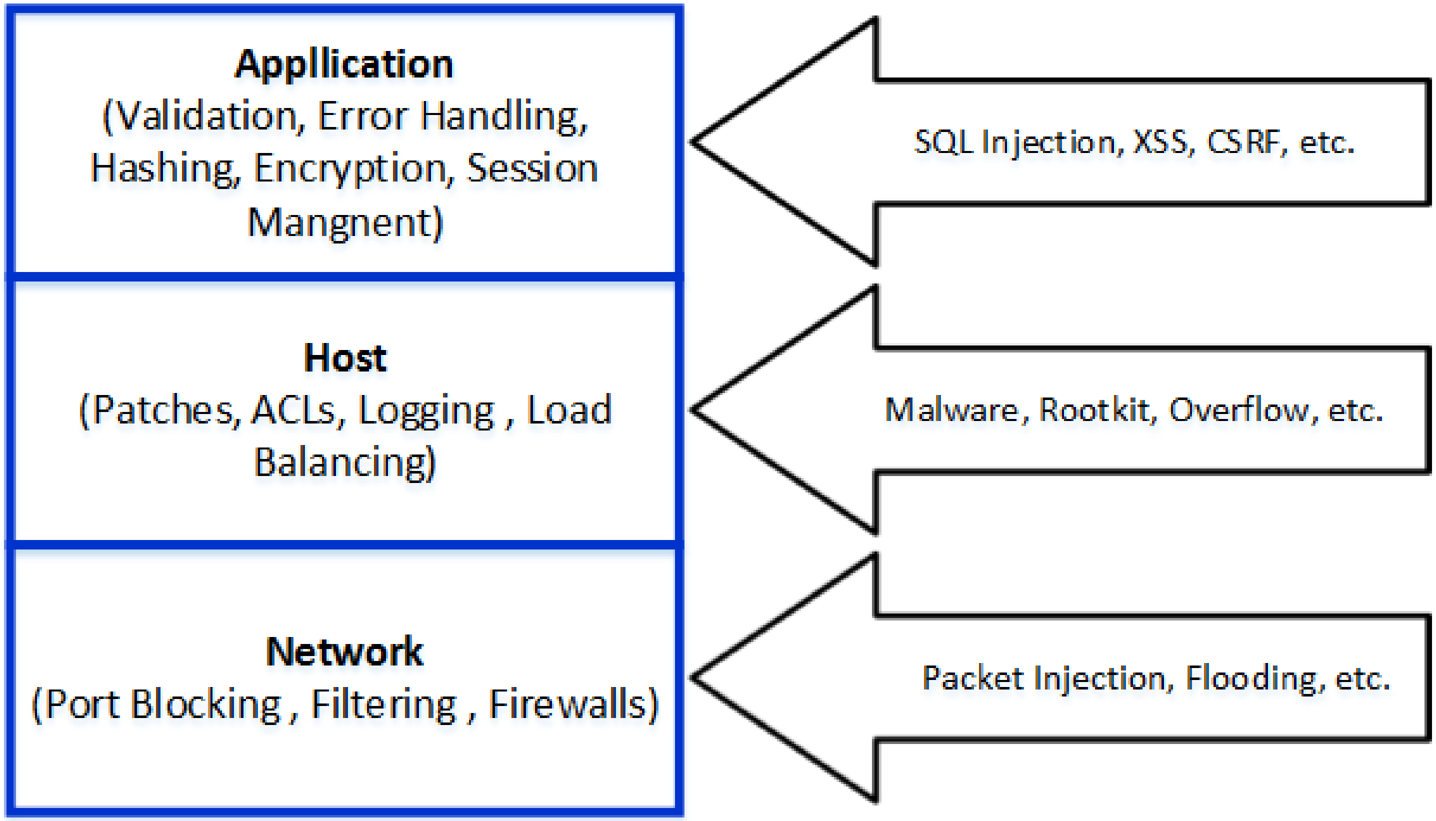
Information in Processing

Information in Transmission

Information in Storage

Holistic Security

Security is qualified by the securing of applications, servers and networks holistically, thus on that point is no weak link



Appllication
(Validation, Error Handling, Hashing, Encryption, Session Mangnent)

SQL Injection, XSS, CSRF, etc.

Host
(Patches, ACLs, Logging , Load Balancing)

Malware, Rootkit, Overflow, etc.

Network
(Port Blocking , Filtering , Firewalls)

Packet Injection, Flooding, etc.

Traditional Security is Reactive

- Perimeter defense (firewalls)
- Intrusion detection (anti-virus)
- Reliance on cryptography
- Penetrate and patch
- Penetration testing

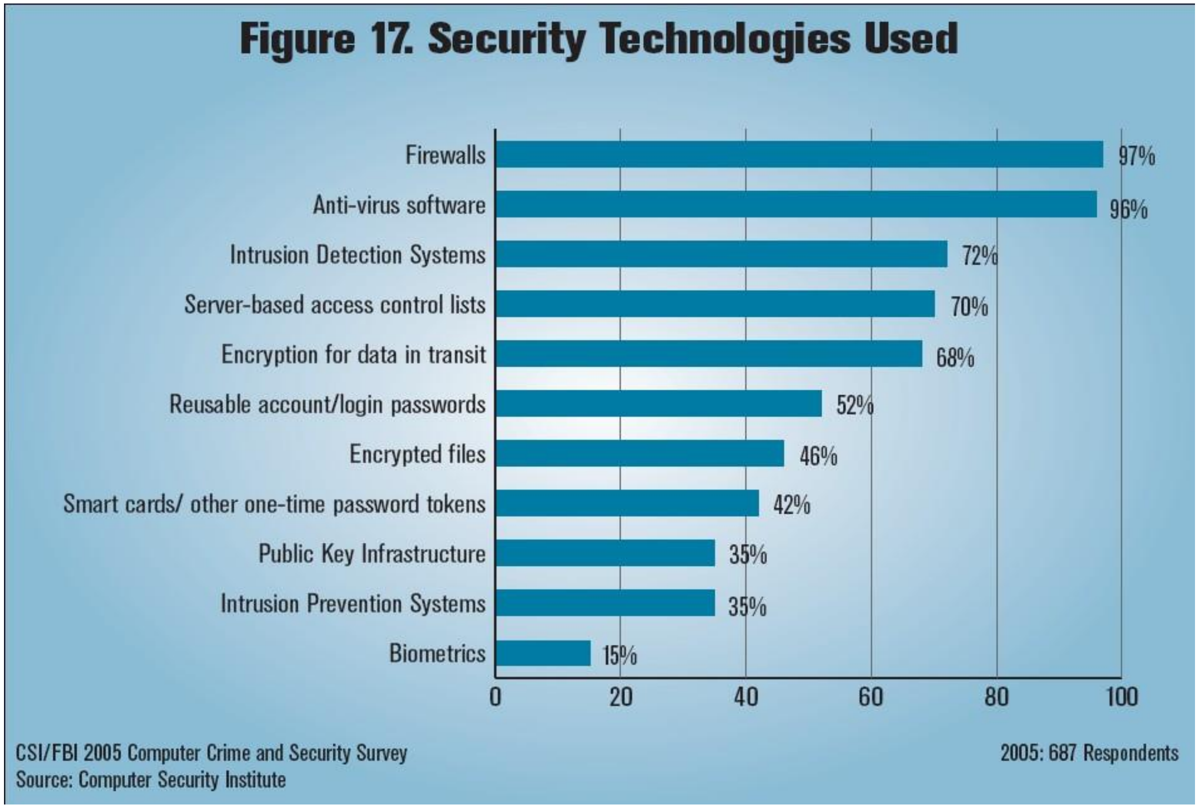


Figure 17. Security Technologies Used

CSI/FBI 2005 Computer Crime and Security Survey
Source: Computer Security Institute

The Problem is Software

- “75% of hacks happen at the application.”
  - Theresa Lanowitz, Gartner Inc.
- “92% of reported vulnerabilities are in apps, not networks.”
  - NIST
- “64% of developers are not confident in their ability to write secure code.”
  - Bill Gates

Trinity of Trouble

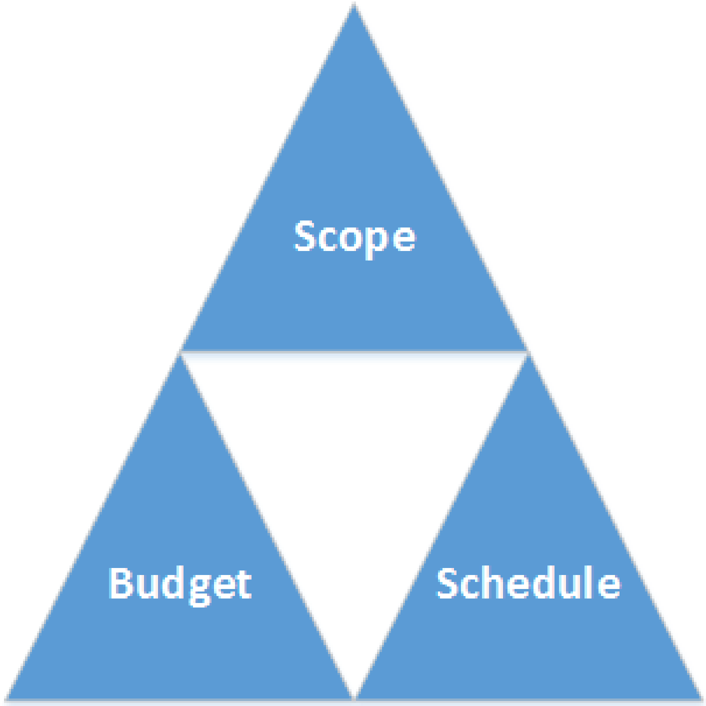
- Connectivity
  - Ubquitious Internet; wireless & mobile computing.
- Complexity
  - Networked, distributed code that can interact with intermediate caches, ad proxies, etc.
- Extensibility
  - Systems evolve in unexpected ways, e.g. web browsers, which support many formats, add- ons, plugins, programming languages, etc.

Security Implementation Challenges

- Iron Triangle Constraints:
  - Scope, Schedule, and Budget

- Security vs. Usability
  - Security must be balanced with usability and performance

- Risk management



Risk management in the context of software security

- Risk management is the balancing act between the protection of IT assets and the cost of implementing software security controls

- NIST SP 800-64 “Security Considerations in the Systems Development Life Cycle (SDLC)”
  - a framework for incorporating security into all phases of the SDLC
- NIST SP 800-30 “Risk Management Guide to Information Technology Systems”

Risk management Challenges

- Risk management for software development is still not maturing
- Software asset values is often subjective

Ways to handle the risk

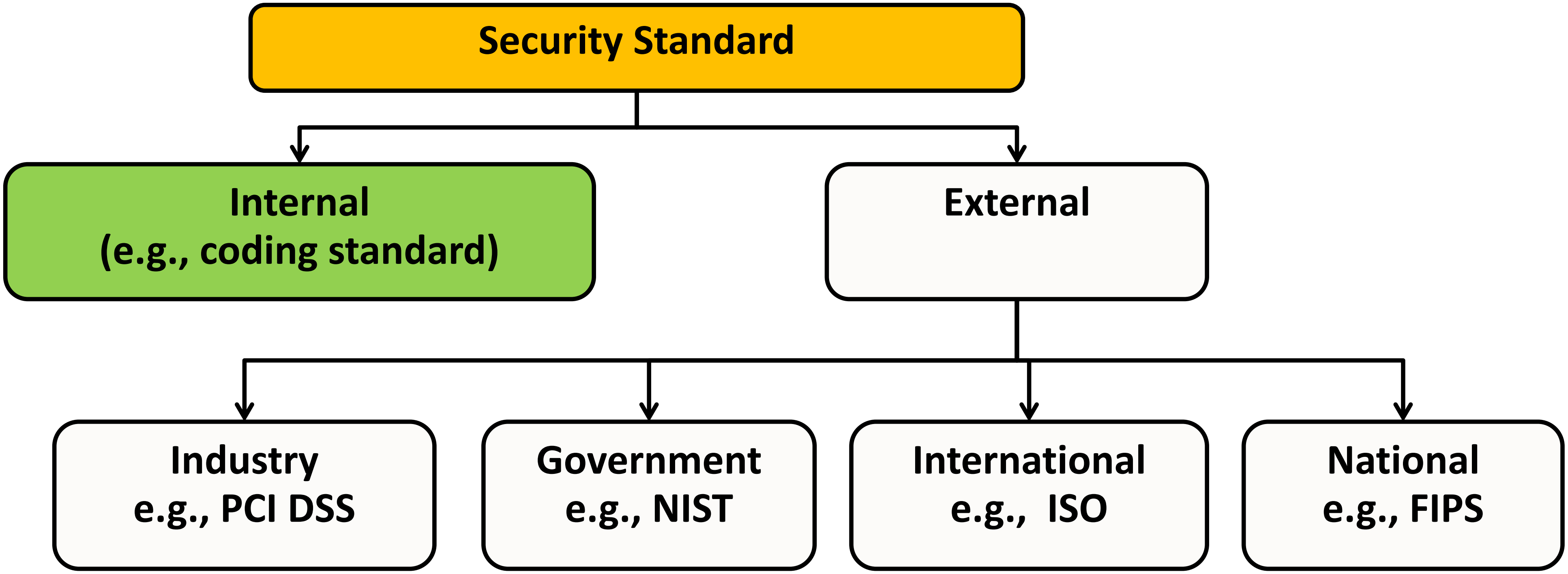
1. Ignore the risk: The risk is left unhandled
   - bad idea
2. Avoid the risk: Discontinue the ecommerce
   - not practical
3. Mitigate the risk: Implement security safeguards
   - may contradict with other regularities
4. Accept the risk: Accept the residual risk
   - must be well documented
5. Transfer the risk: Buying insurance and using disclaimers
   - software security insurance is not very common

SSE Objectives

Secure Software Engineering Objectives

1. **Dependability**
   - software functions only as intended;
2. **Trustworthiness**
   - No exploitable vulnerabilities or malicious logic exist in the software;
3. **Resilience**
   - If compromised, damage will be minimized, and it will recover quickly to an acceptable level of operating capacity;
4. **Conformance**
   - to requirements and applicable standards and procedures.

Security Standards



PCI DSS: Payment Card Industry Data Security Standard

-*Requirement 6:* Develop and maintain secure systems and applications
https://www.pcisecuritystandards.org/pci_security/

NIST: National Institute of Standards and Technology
ISO: International Organization for Standardization
FIPS: Federal Information Processing Standards

Security Standard

NIST Standards

- SP 800-12: An Introduction to Computer Security: the NIST Handbook
- SP 800-14: Generally Accepted Principles and Practices for Security IT Systems
- SP 800-18: Guide for developing Security Plans for Federal Systems
- SP 800-27: Engineering Principles for Information Technology Security
- SP 800-30: Risk Management Guide for IT
- SP 800-61: Computer Security Incident Handling Guide
- SP 800-64: Security Considerations in the Information Systems Development Life Cycle
- SP 800-100: Information Security Handbook: A Guide for Managers

FIPS standards

- FIPS 140: Security Requirement for Cryptographic Modules
- FIPS 186: Digital Signature Standard
- FIPS 197: Advanced Encryption Standard
- FIPS 201: Personal Identity Verification (PIV) of Federal Employees and Contractors

ISO Standards

- ISO/IEC 15408 – Evaluating Criteria for IT Security (Common Criteria)
- ISO/IEC 15408- Standard and Software Security
- ISO/IEC 21827:2008 – Systems Security Engineering Capability Maturity Model® (SSE-CMM®)
- ISO/IEC 25000:2005 – Software Engineering Product Quality
- ISO/IEC 27000:2009 – Information Security Management System (ISMS) Overview and Vocabulary
- ISO/IEC 27001:2005 – Information Security Management Systems Requirements
- ISO/IEC 27002:2005/Cor1:2007 – Code of Practice for Information Security Management
- ISO/IEC 27000:2009 – Information Security Management System (ISMS) Overview and Vocabulary
- ISO/IEC 27005:2008 - Information Security Risk Management

Secure Software Certifications and Training

- (ISC)² Certified Secure Software Lifecycle Professional (CSSLP)
  - https://www.isc2.org/Certifications/CSSLP
- SANS GIAC Secure Software Programmer
  - https://software-security.sans.org/certification
- WebGoat
  - maintained by OWASP
  - designed to teach web application security lessons

Security in SDLC

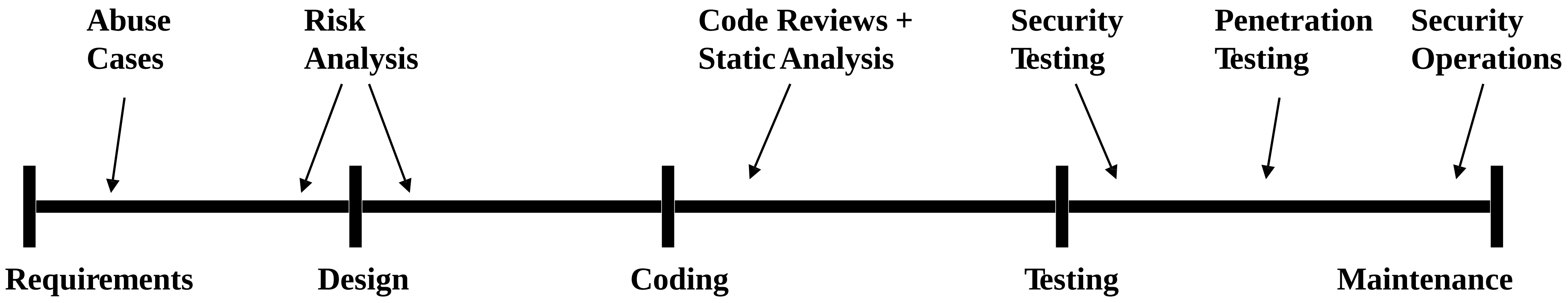
- Touchpoints
  - http://www.swsec.com/resources/touchpoints/
- MS SDL - (Microsoft Secure Development Lifecycle)
  - https://www.microsoft.com/en-us/SDL
- SAMM - The **S**oftware **A**ssurance **M**aturity **M**odel
  - https://www.owasp.org/index.php/OWASP_SAMM_Project
- TSP-Secure (Team Software Process for Secure Software Development)
  - http://www.sei.cmu.edu/engage/collaborate/cases/tsp-secureorigins.cfm
- SAFECode
  - http://www.safecode.org/

Secure Software Development Lifecycle (SSDL)

- **SSDL** must have in a software development projects
- The SSDL takes a **holistic** approach to secure software development
  - Secure Software Concepts
  - Secure Software Requirements
  - Secure Software Design
  - Secure Software Implementation/Coding
  - Secure Software Testing
  - Software Acceptance
  - Software Deployment, Operations, Maintenance, and Disposal

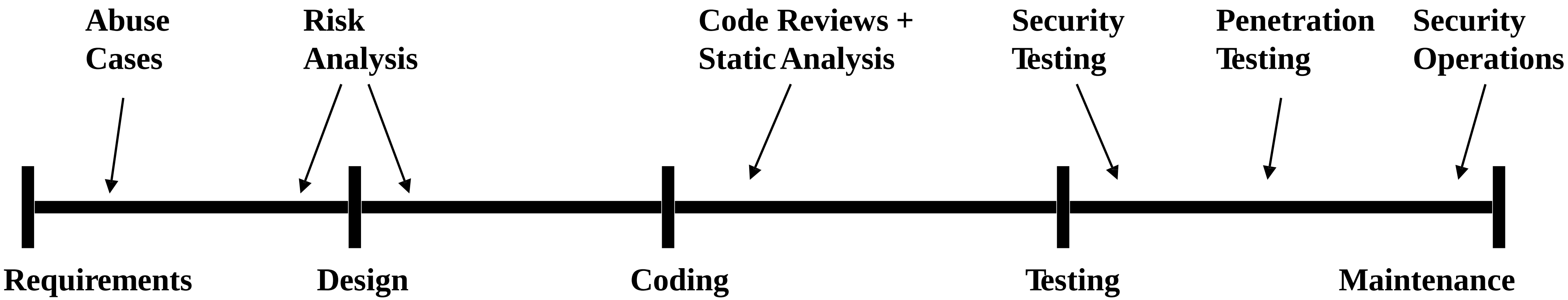
Software Security Practices

1. Code Reviews
2. Risk Analysis
3. Penetration Testing
4. Security Testing
5. Abuse Cases
6. Security Operations



Code Reviews

- Fix implementation bugs, not design flaws.
- Benefits of code reviews
  1. Find defects sooner in the lifecycle.
  2. Find defects with less effort than testing.
  3. Find different defects than testing.
  4. Educate developers about security flaws.



Abuse Cases

Risk Analysis

Code Reviews + Static Analysis

Security Testing

Penetration Testing

Security Operations

Requirements

Design

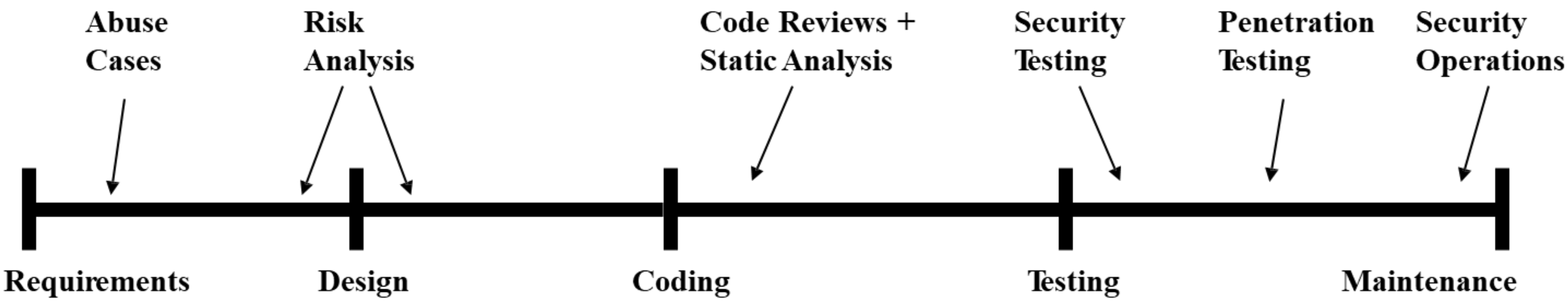
Coding

Testing

Maintenance

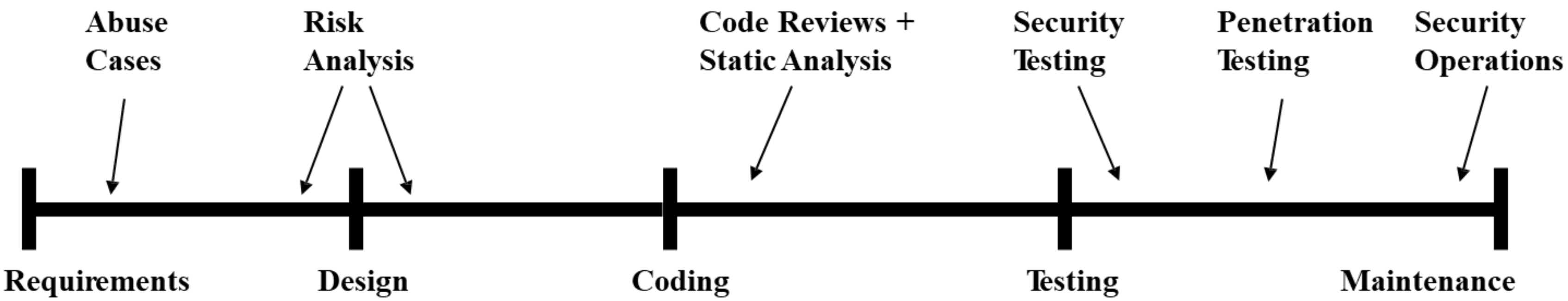
Architectural Risk Analysis

- Fix design flaws, not implementation bugs.
- Risk analysis steps
  1. Develop an architecture model.
  2. Identify threats and possible vulnerabilities.
  3. Develop attack scenarios.
  4. Rank risks based on probability and impact.
  5. Develop mitigation strategy.
  6. Report findings



Penetration Testing

- Test software in deployed environment.
- Allocate time at end of development to test.
  - Often time-boxed: test for *n* days.
  - Schedule slips often reduce testing time.
  - Fixing flaws is expensive late in lifecycle.
- Penetration testing tools
  - Test common vulnerability types against inputs.
  - Fuzzing: send random data to inputs.
  - Don’t understand application structure or purpose.



Abuse Cases

Risk Analysis

Code Reviews + Static Analysis

Security Testing

Penetration Testing

Security Operations

Requirements

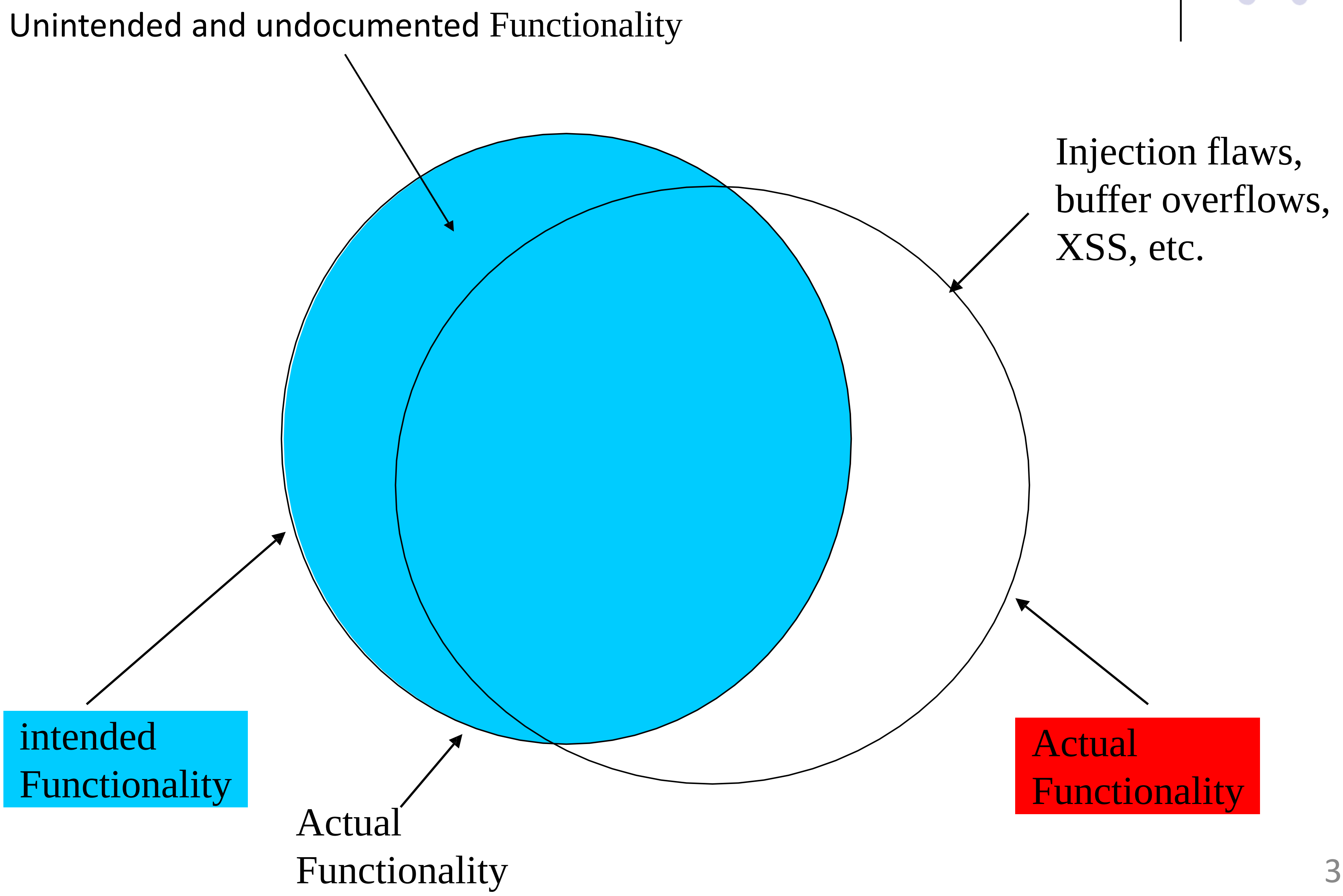
Design

Coding

Testing

Maintenance

Security Testing



Unintended and undocumented Functionality

Injection flaws, buffer overflows, XSS, etc.

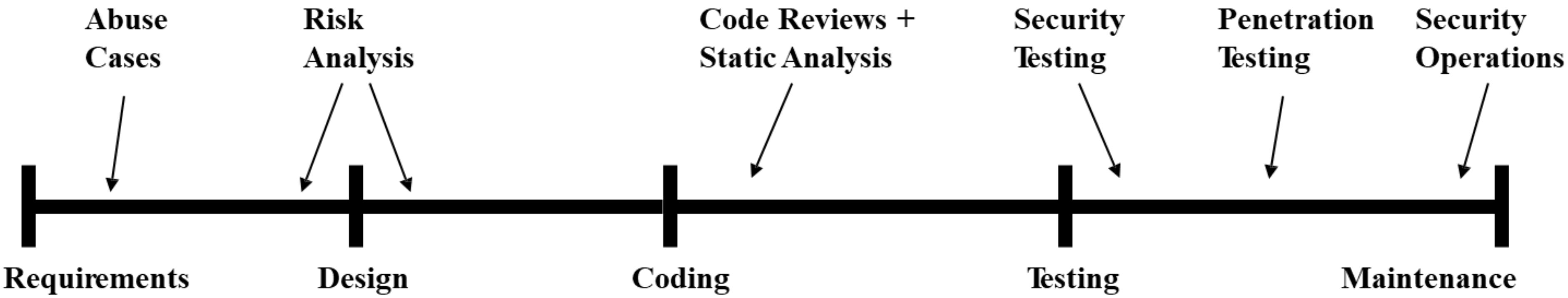
intended Functionality

Actual Functionality

Actual Functionality

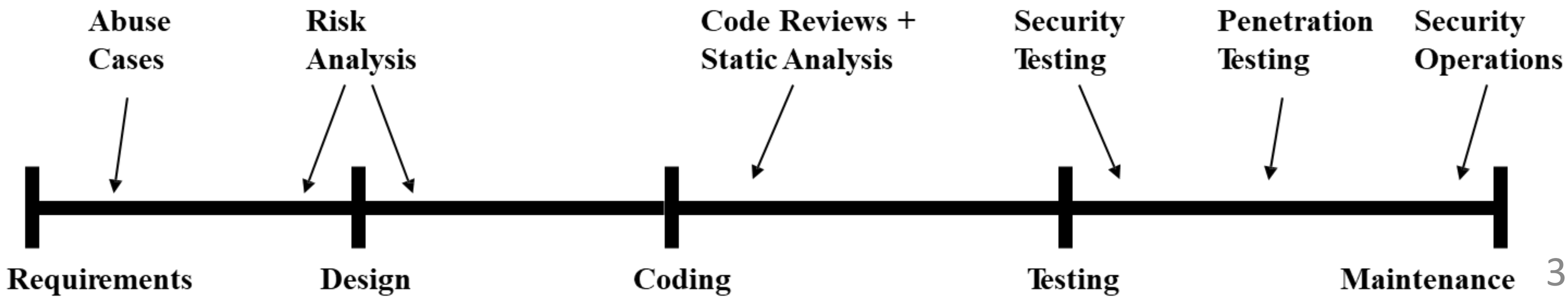
Security Testing

- Two types of testing
  - **Functional:** verify security mechanisms.
  - **Adversarial:** verify resistance to attacks generated during risk analysis.
- Different from traditional penetration testing
  - White box.
  - Use risk analysis to build tests.
  - Measure security against risk model.



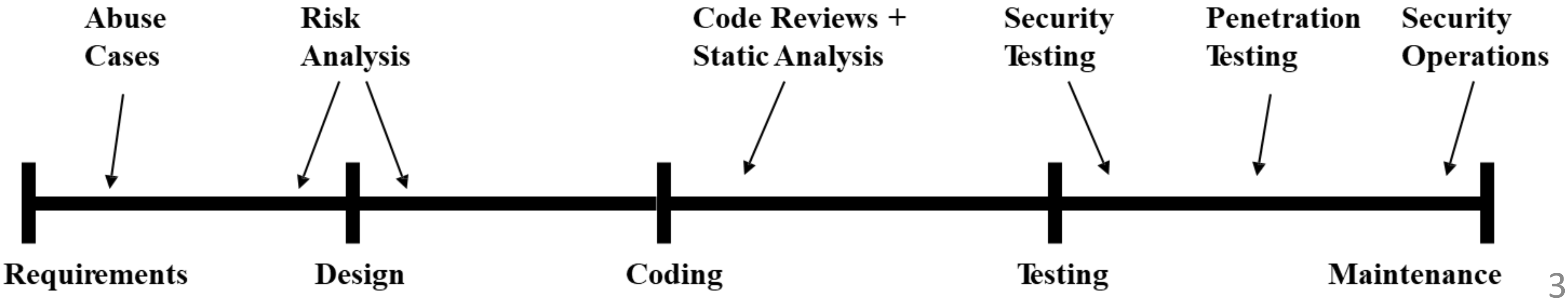
Abuse Cases

- Anti-requirements
  - Think about what software should not do.
- A use case from an adversary’s point of view.
  - Obtain Another User’s CC Data.
  - Alter Item Price.
  - Deny Service to Application.
- Developing abuse cases
  - Informed brainstorming: attack patterns, risks.



Security Operations

- User security notes
  - Software should be secure by default.
  - Enabling certain features may have risks.
  - User needs to be informed of security risks.
- Incident response
  - What happens when a vulnerability is reported?
  - How do you communicate with users?
  - How do you send updates to users?



Key Points

- Touchpoints
  - Code Reviews
  - Risk Analysis
  - Penetration Testing
  - Security Testing
  - Abuse Cases
  - Security Operations

- Components of Security
  - Confidentiality
  - Integrity
  - Availability

Summary

- Software security is no longer can be on the sidelines
- Security needs to be included in software development life cycle from the beginning
- Software risk management has special challenges
- Policies, standards, and common methodologies for best practices and framework were covered
- Abstract security models software security were introduced